**Московский государственный технический университет им. Н.Э. Баумана**

Факультет «Радиотехнический»

Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Базовые компоненты интернет-технологий» Отчёт по рубежному контролю №2

Вариант №15

|  |  |
| --- | --- |
| Выполнил: | Проверил: |
| студент группы РТ5-31Б | преподаватель каф. ИУ5 |
| Мицкевич Владислав | Гапанюк Ю.Е. |
| Подпись и дата: | Подпись и дата: |

Москва, 2021 г

# Описание задания

Рубежный контроль представляет собой разработку тестов на языке Python.

1. Проведите рефакторинг текста программы рубежного контроля №1 таким образом, чтобы он был пригоден для модульного тестирования.
2. Для текста программы рубежного контроля №1 создайте модульные тесты с применением TDD - фреймворка (3 теста).

# Текст программы

**RK2**

class File:

    def \_\_init\_\_(*self*, *id\_file*, *name\_file*, *size\_file*, *file\_extension*, *id\_directory*):

*self*.name\_file = *name\_file*

*self*.size\_file = *size\_file*

*self*.file\_extension = *file\_extension*

*self*.id\_file = *id\_file*

*self*.id\_directory = *id\_directory*

class Directory:

    def \_\_init\_\_(*self*, *name\_directory*, *size\_directory*, *id\_directory*):

*self*.name\_directory = *name\_directory*

*self*.size\_directory = *size\_directory*

*self*.id\_directory = *id\_directory*

    def \_\_repr\_\_(*self*):

        return f'{*self*.size\_directory} {*self*.name\_directory} {*self*.size\_directory}'

class DirFile:

    def \_\_init\_\_(*self*, *id\_directory*, *id\_file*):

*self*.id\_directory = *id\_directory*

*self*.id\_file = *id\_file*

files = [File(7, "Lab1", 101, "docx", 1), File(5, "Lab2", 150, "docx", 1), File(120, "Lab3", 98, "docx", 1),

        File(4, "Math", 700, "pdf", 3), File(27, "Python\_Book", 850, "pdf", 3), File(2, "It", 1500, "djvu", 3),

        File(145, "Tester", 160, "py", 7)]

directories = [Directory("Labs", 340, 1), Directory("Film", 0, 45), Directory("BookForLabs", 3050, 3), Directory("Pycharm\_programmers", 160, 7)]

one\_to\_many = [(x,[y for y in files if y.id\_directory == x.id\_directory]) for x in directories]

many\_to\_many = [DirFile(1, 7), DirFile(1, 5), DirFile(1, 120), DirFile(1, 5416), DirFile(3, 4), DirFile(7, 145), DirFile(3, 27)]

def test1():

*#«Отдел» и «Сотрудник» связаны соотношением один - ко - многим.Выведите список всех*

*#отделов, у которых в названии присутствует слово «отдел», и список работающих в них сотрудников.*

    a = str()

    for dir, fil in one\_to\_many:

        if dir.name\_directory.find('Labs') != -1:

            a += dir.name\_directory + ": "

            for x in fil:

                a += x.name\_file + " "

            a += "\n"

    return a

def test2():

*#«Отдел» и «Сотрудник» связаны соотношением один - ко - многим.Выведите список отделов со средней зарплатой*

*#сотрудников в каждом отделе, отсортированный по средней зарплате.Средняя зарплата должна быть округлена до 2 знака после запятой*

    def func(*lst*):

        return *lst*[1]

    lst = list()

    for dir, fil in one\_to\_many:

        sr = 0

        sum = 0

        if len(fil) != 0:

            for x in fil:

                sum += x.size\_file

            sr = round(sum / len(fil), 2)

            lst.append([dir.name\_directory, sr])

        else:

            lst.append([dir.name\_directory, 0])

    a = sorted(lst, *key*=func)

    return(a)

def test3():

*#«Отдел» и «Сотрудник» связаны соотношением многие - ко - многим.Выведите список всех сотрудников, у*

*#которых фамилия начинается с буквы «А», и названия их отделов.*

    a = ''

*#Отдел» и «Сотрудник» связаны соотношением многие-ко-многим. Выведите список всех сотрудников,*

*# у которых фамилия начинается с буквы «А», и названия их отделов.*

    for x in many\_to\_many:

        for y in files:

            if x.id\_file == y.id\_file:

                if y.name\_file[0] == 'L':

                    a += y.name\_file + "\t"

                    for z in directories:

                        if z.id\_directory == x.id\_directory:

                            a += z.name\_directory + "\n"

    return a

def main():

    print(test1())

    print(test2())

    print(test3())

main()

**test.py**

import unittest

import РК2

class test(unittest.TestCase):

    def test\_tasks(*self*):

*self*.assertEqual(РК2.test1(),'Labs: Lab1 Lab2 Lab3 \nBookForLabs: Math Python\_Book It \n')

*self*.assertEqual(РК2.test2(), [['Film', 0], ['Labs', 116.33], ['Pycharm\_programmers', 160.0], ['BookForLabs', 1016.67]])

*self*.assertEqual(РК2.test3(),'Lab1\tLabs\nLab2\tLabs\nLab3\tLabs\n')

if \_\_name\_\_ == "\_\_main\_\_":

    unittest.main()

# Пример работы программы

